Sketchy Illustrations for Presenting the Design of Interactive CSG

Marc Nienhaus, Florian Kirsch, Jürgen Döllner
University of Potsdam, Hasso Plattner Institute
{nienhaus@hpi.uni-potsdam.de, kirsch@hpi.uni-potsdam.de, doellner@hpi.uni-potsdam.de}

Abstract

Illustrating in a sketchy manner is essential to communicate visual ideas and can be used to present and reconsider drafts and concepts in product design.

This paper introduces a real-time illustration technique that sketches the design and spatial assembly of CSG models. The illustration technique generates a graphical decomposition of the CSG model into disjunctive layers to extract 1) the perceptually important edges that outline the model's outer and inner features and 2) the surface shading of the outer and inner faces. Then, the technique applies uncertainty to these layers to simulate a sketchy effect. Finally, the technique composes the sketched layers in depth-sorted order while ensuring a correct depth behavior in the frame buffer.

Because the sketchy illustrations are frame-to-frame coherent, the technique can be used as a tool for interactive presentation and reconsideration of the design and spatial assembly of CSG models.

Keywords: **Illustrative Visualization, NPR**.

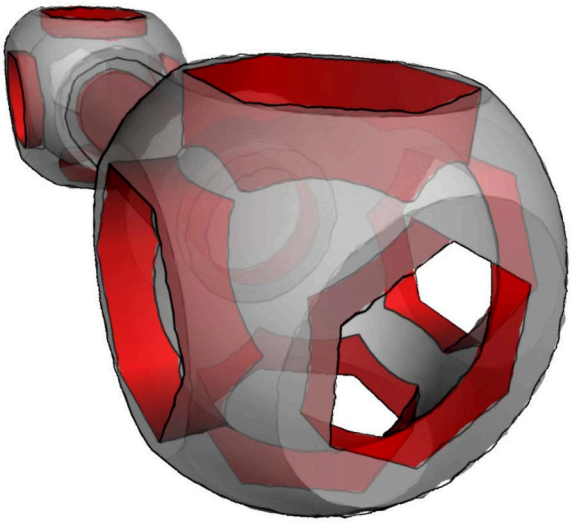

Figure 1: A sketchy illustration of the spatial assembly of a CSG model, a spanner.

1. Introduction

Rendering in a "sketchy" manner is of vital importance for communicating visual ideas and for illustrating the preliminary state of a draft or concept, especially in application areas such as product design.

Common photorealistic renditions are often less efficient in proposing ideas and concepts, because they imply the impression of finality and correctness. Photorealistic renderings thus reduce one's ability to rethink enhancements and modifications. In contrast, sketches communicate visually and can therefore be more helpful when dealing with renditions. In particular, sketches encourage the exchange of ideas when people are reconsidering drafts; sketches express uncertainty and suggest work in progress [21] . In fact, hand-drawn sketches are still an integral part of the development process in architectural or product design.

Outlining and enhancing visible and occluded features in drafts of mechanical parts are essential techniques for illustrating complex aggregate objects and for illustrating the position, layout, and relationships of their components.

Constructive Solid Geometry (CSG) represents a fundamental concept for modeling and designing 3D solids and complex aggregate objects such as mechanical parts. However, the graphical representations synthesized by interactive, image-based CSG rendering algorithms generally portray the CSG models by simple shaded geometries represented only by the outer surface of their solids. As a result, the interior composition of the CSG models is not visible and, thus, hardly perceivable, i.e., the entire assembly becomes increasingly difficult to understand. Furthermore, a lack of sketchiness hinders active participation into a presented design and, thus, hinders reconsideration and decision making.

Non-photorealistic rendering (NPR) has become a core discipline in computer graphics [5] [23] . It is concerned with the generation of vivid and expressive depictions that assist visual perception. NPR has been realized as means for visualizing illustratively and gains increased acceptance [4] , for instance, in medical or scientific visualization [17] [20] .

Proceedings of the Information Visualization (IV'06)
0-7695-2602-0/06 $20.00 © 2006 IEEE

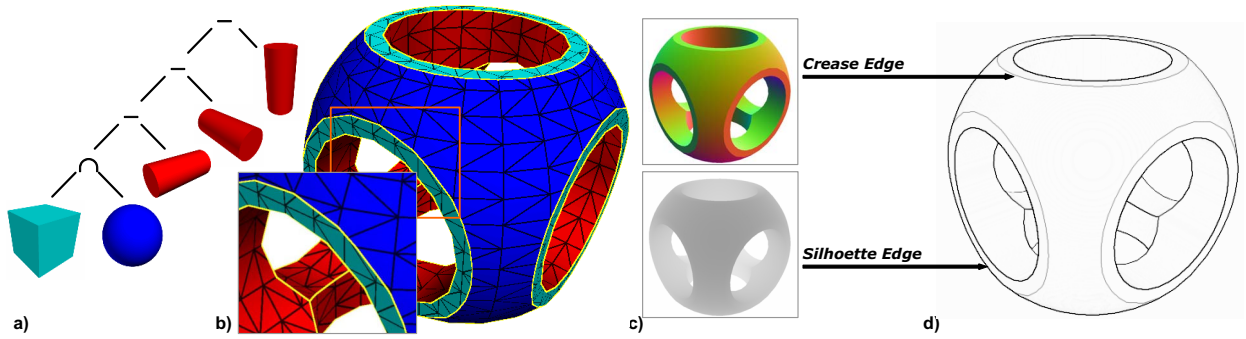

Figure 2: The CSG tree (a) results in the CSG model of the widget. Perceptually important edges of clipped polygonal geometries do not correspond to the polygonal edges (b). Discontinuities in the normal and depth buffer (c) form perceptually important edges in the edge map (d).

We present an NPR illustration technique that (1) outlines the spatial assembly of a CSG model to provide spatial insight into its complex assembly and let understand it as a whole and (2) sketches these outlines to present the CSG model's design as a draft for the purpose of reconsideration (Fig 1).

2. Interactive, Image-based CSG

CSG modeling means defining 3D geometry as the result of set operations ($\cap$, $\cup$, -), applied to basic, closed 3D primitives or to other CSG geometry defined in this way. The CSG tree represents the fundamental structure for specifying a CSG models (Fig. 2a and 2b).

Goldfeather et al. developed one of the first algorithms for image-based rendering of CSG [3] . He introduced the normalization of arbitrary CSG trees into an equivalent union-of-partial-product form. The visibility of the partial products can be effectively determined by image-based graphics hardware operations. Wiegand describes a rendering technique that implements the algorithm of Goldfeather et al. using OpenGL [24] . He used an emulation of two z-buffers, one for calculating the visibility of a single primitive and another to combine the results. Kirsch and Döllner improve this approach by a real-time capable rendering technique that transfers visibility information using color textures [9] . Kirsch also introduces order-independent transparent illustrations of interactive CSG [8] and Nienhaus et al. present an illustration technique that is capable of conveying a CSG model's internal layout [15] . We make substantial use of their previous work.

3. Perceptually Important Edges

Outlines, particularly edges where the visibility of the surface changes (a.k.a. silhouettes) of 3D models are one of the strongest visual cues [11] , "important for figure-to-background distinctions" [7] , and "communicate the essence of shape" [6] . We denote edges that let one perceive shape as *perceptually important edges*.

3D models typically consist of multiple meshes that possibly intersect one another. In case of CSG models, part of the surface may be clipped, e.g., by auxiliary geometry for the purposes of modeling (Fig. 2a). In both cases, additional polygonal edges are generally not inserted into the polygonal 3D models' representations. Essentially, perceptually important edges represent no subset of the meshes' polygonal edges. Thus, a classification based on polygonal edges is not sufficient to define perceptually important edges unambiguous.

3.1 Edge Classification

Our classification of perceptually important edges considers the visibility and orientation of a model's surface:

- A junction where two polygons adjoin, where one polygon is visible and the other one is occluded along the junction, represents a *silhouette edge*. That is, the visibility of the surface changes along the junction.
- The boundary of a polygon where no polygon adjoins represents a *border edge*.
- A junction where two polygons adjoin, where both polygons are visible along the junction (i.e., the visibility of the surface remains unchanged), and form a certain angle above some threshold represents a *crease edge*.

In case of solid models such as CSG models no border edges exist. Figure 2b illustrates the perceptually important edges of a CSG model (yellow) and the polygonal edges of the underlying meshes (black).

3.2 Image-Space Edge Detection

Object-space edge-detection algorithms [7] can hardly determine perceptually important edges efficiently without re-meshing a CSG model's geometry. In contrast, image-space edge-detection algorithms can extract these edges efficiently. Such algorithms are based on the G-Buffer concept [19] . The edge-enhancement algorithm [12] extracts silhouette edges by detecting discontinuities in a depth buffer and crease edges by detecting discontinuities in a normal buffer (Fig. 2c). To generate these buffers, we render z-values and encoded per-fragment normal values directly into textures. Then,

we texture a screen-aligned quad using these textures. Sampling neighboring texture values lets us extract discontinuities in both buffers which results in intensity values. The assembly of intensity values constitutes perceptually important edges that we store into a single texture, called the *edge map* (Fig. 2d).

4. Rendering Sketchy

Edges as well as the surfaces shading derived from CSG models are considered for rendering in a sketchy manner. Besides the edge map, the rendering technique also renders the surface colors of the CSG model into a texture referred to as the *shade map* (Fig. 3b).

We look for uncertainty values that let one sketch edges and colors pseudo-randomly in image space to simulate the effect of "sketching on a flat surface". Perlin introduced a pseudo-random noise function that serves as a primitive for controllable noise [16] . The stochastic patterns produced by the Perlin noise function maintain spatial coherence, that is, noise values calculated for adjacent input parameters are correlated to one another.

Again, we texture a canvas-filling quad with the edge and shade maps as input. Furthermore, we apply a noise texture, whose values represent uncertainty values.

The noise texture serves as an offset texture to implement a dependent texture access for the edge and shade maps. That is, its texture values perturb slightly the texture coordinates of each fragment of the quad that accesses the edge and shade maps

Furthermore, we control the amount of perturbation by a user-defined 2×2 matrix. The uncertainty values of the noise texture are multiplied by that matrix to weight all these values uniformly before translating the texture coordinates. Figure 3a illustrates the perturbation of the texture coordinates that access the shade map using the degree of uncertainty. Here, a texture coordinate, which would yield a texture value that does not correspond to the 3D shape, accesses a region that corresponds to the shape.

In order to enhance the sketchiness effect, the texture coordinates for accessing the edge and shade maps are perturbed differently using two different 2×2 matrices. This results in different degrees of uncertainty for each map that let shift the texture coordinates of the edge map and the texture coordinates of the shade map in contrary directions. Figure 3b´ shows the shade map and Figure 3c´ shows the edge map after uncertainty has been applied.

Finally the resulting texture values of both the edge and the shade maps are combined. The intensity values derived from perturbing the edge map are multiplied by the color values derived from perturbing the shade map.

Figure 3e shows the sketchy illustration of the CSG model. Because neighboring uncertainty values (based on Perlin noise) are correlated in image space, a continuous sketchy boundary and frame-to-frame coherence through interaction and animation can be maintained.

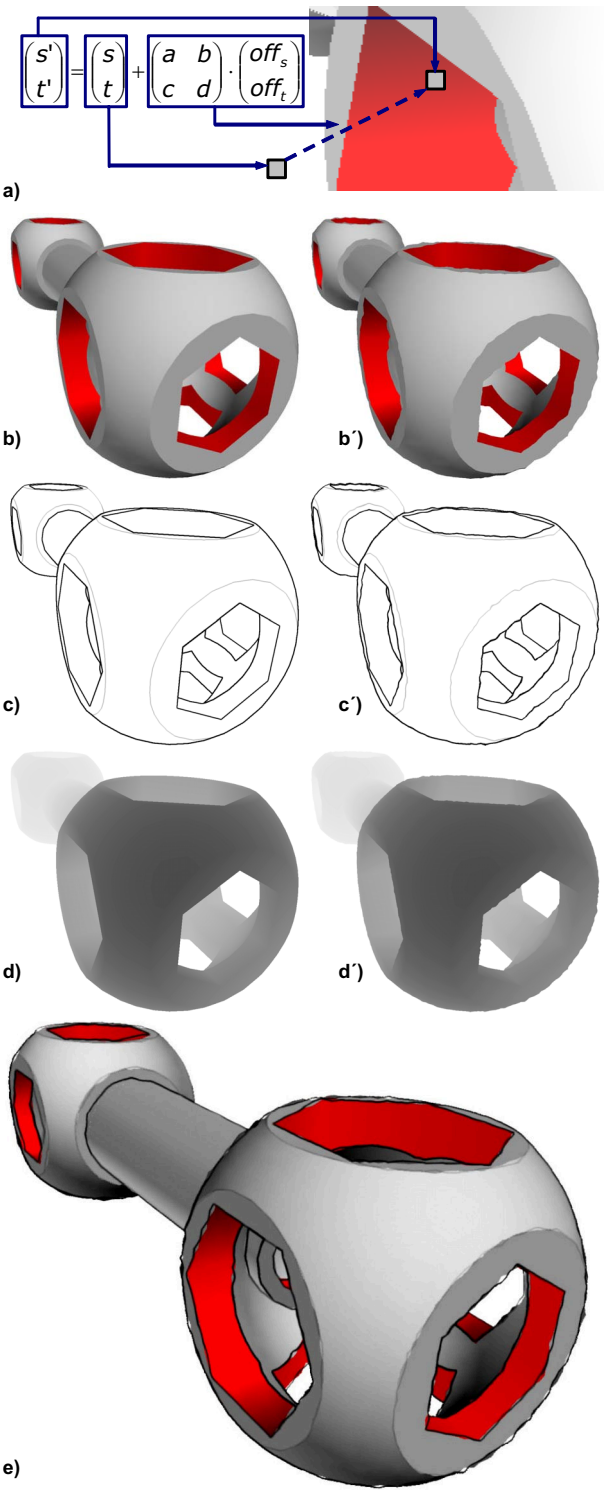

Figure 3: Sketchiness is achieved by perturbing texture coordinates in image space using uncertainty values (a). (b) The perturbed shade map (c) the perturbed edge map and (d) the perturbed depth map let implement the (e) sketchy illustrations of the CSG model.

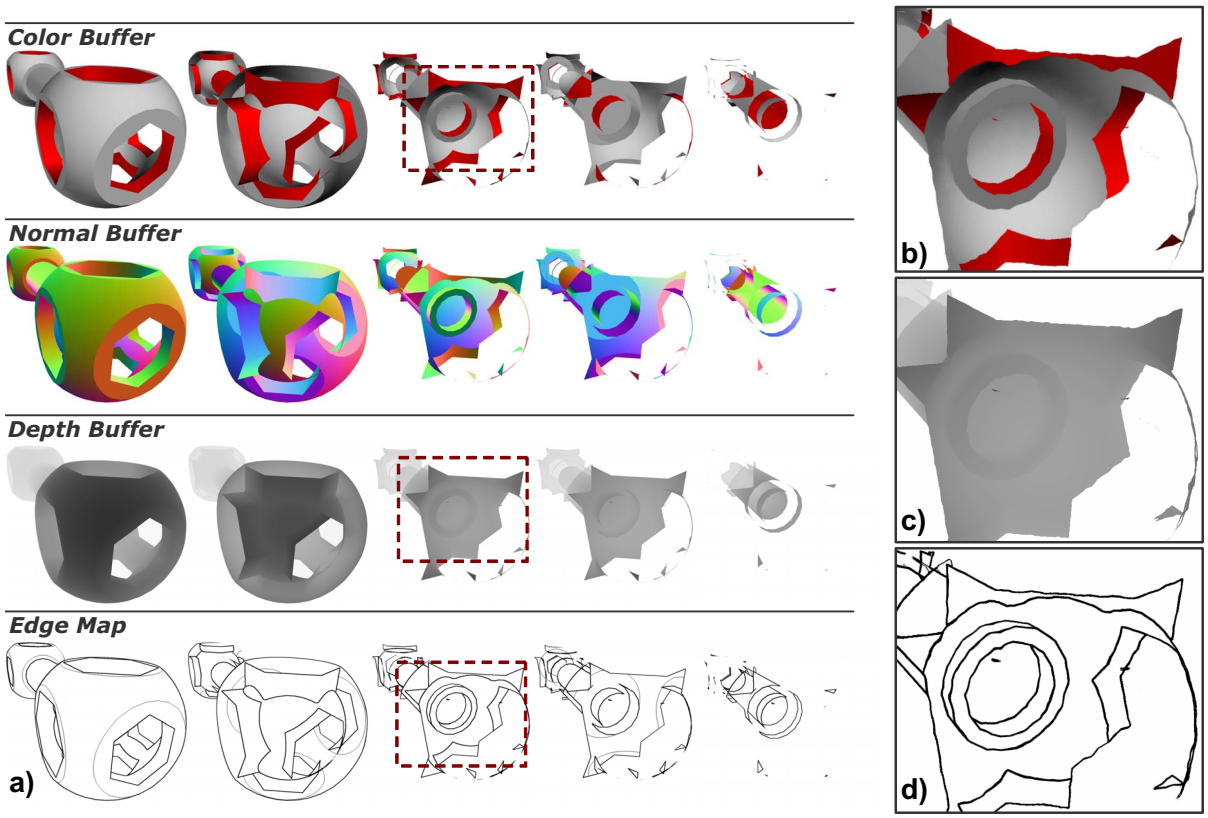

Figure 4: (a) Color buffer and normal buffer are generated and then the edge maps are constructed for each layer. Perturbing the contents of (b) the shade map, (c) the depth map, and (d) the edge map for each layer enables one to render spatial assemblies in a sketchy manner.

4.1 Adjusting Depth Values

Up to now, a screen-aligned quad has been textured with a sketchy illustration. This has the following shortcomings:

- Z-values of the original geometry are replaced by the z-values of the quad.
- No z-values of the original 3D geometry are present in regions beyond the geometric boundary when uncertainty has been applied.

Thus, the sketchy illustration cannot interact correctly with other objects in the 3D scene.

To overcome these shortcomings, the sketchy illustration is rendered as depth sprite allowing for previous perturbations. For this, the depth buffer (already used for edge detection) is captured as textures high-precision depth map (Fig. 3d). Then the canvas-filling quad gets also textured with the depth map, whereas both the previous perturbations are applied to its texture coordinates. The minimum value of both texture values produces the final z-value of the quad's fragment for depth testing. Figure 3d′ shows the result of both perturbations applied to the depth map. The interior region of the perturbed depth map matches the combination of the interior regions of both the perturbed edge map and the perturbed shade map.

Modifying depth sprite rendering thus allows one to adjust the z-values of a screen-aligned quad that is textured by the 3D model's sketchy depiction. In this way, the sketchy illustration behaves correctly with respect to depth, that is, the z-buffer remains in a correct state with respect to the geometry of the CSG model.

5. Illustrating Spatial Assemblies

Outlining and enhancing visible and occluded features enable illustrative depictions of the spatial assemblies of CSG models. Visible perceptually important edges enhance the outlines of visible features whereas perceptually important edges that are not directly seen by the virtual camera outline occluded features, e.g., the internal structure of the CSG model.

5.1. Graphical Decomposition

A graphical decomposition of the CSG models consists of a set of disjunctive layers given in depth-sorted order. Each layer represents a view on the model corresponding to a unique depth complexity. The first layer shows the directly visible surface of the CSG model. The second layer shows its internal parts as if the first layer has been peeled away. Yet, this second layer shows only those parts that are then nearest to the viewer

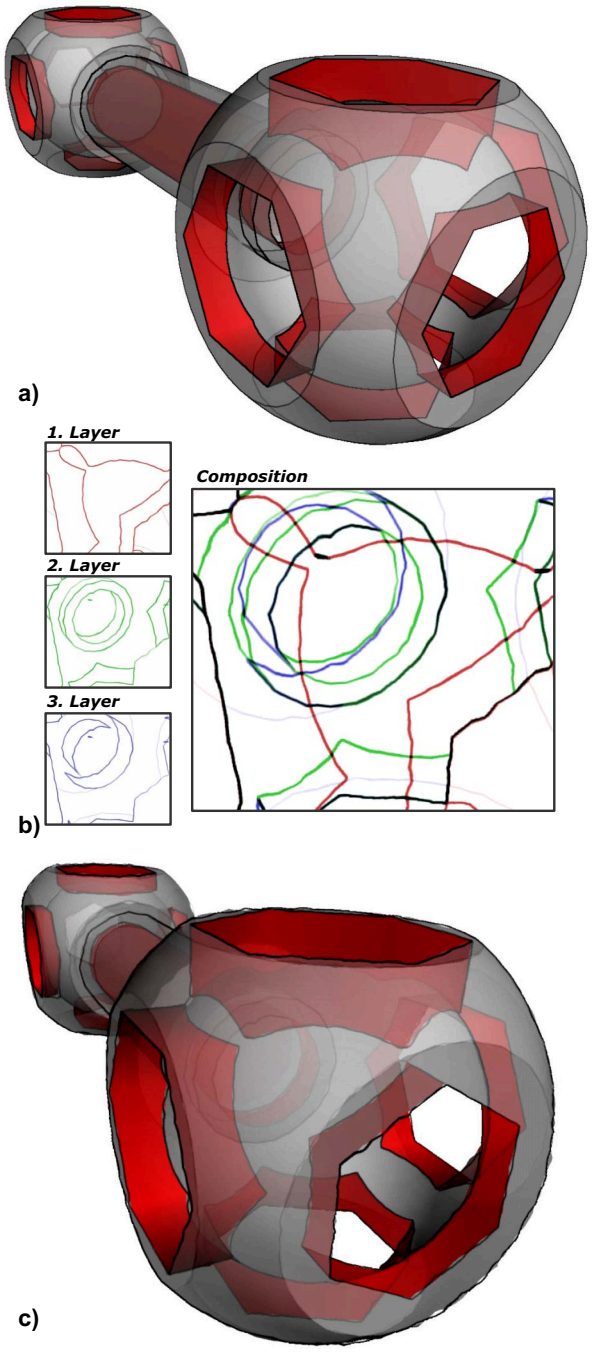

Figure 5: (a) The illustration of the spatial assembly of the CSG model. (b) Repeated edges in edge maps of consecutive depth layer match to one another even though uncertainty has been applied. (c) The sketchy illustration of the CSG model's design.

and does not provide a deeper insight into the assembly. Thus, this second layer must also be peeled away to show the next layer to step deeper and so forth.

Technically, the decomposition is generated by a multipass rendering process. In each rendering pass, two depth tests are performed on a per fragments basis whereas the depth buffer of the previous layer is used for one test ([1] ,[2] ,[13]).

This general approach does not map directly to image-based CSG rendering, which substantially uses the fact that CSG primitives are only partially visible and calculates the visible areas in image-space. Instead of linearly traversing all CSG primitives for each layer, CSG calculations are required for each layer.

For inner layers (i.e., layers corresponding to a depth complexity greater than 1) a CSG calculation has to be performed twice for each CSG primitive: Once to determine those visible parts of the surface of the CSG product that faces towards the camera and once to determine the visible parts of the surface that faces in the opposite direction. Potentially visible parts are determined with respect to the depth buffer of the previous depth layer, i.e., only fragments behind the previous layer are considered to be potentially visible, which finally let determine the visible parts of a CSG product [8] .

5.3. Extracting Visible and Occluded Edges

For each layer the graphical decomposition provides both the current depth and color buffers as textures for later usage (Fig. 4a). Furthermore, today's graphics cards allow populating multiple color buffers at once [10] [18] . Thus, encoded normal values can be passed to an additional color buffer in order to synthesize a normal buffer for each layer.

Given both the depth and the normal buffers allows generating an edge map for each layer (see 3.2). As a result, the visible perceptually important edges are stored in the edge map of the first layer and the occluded perceptually important edges are stored in edge maps of subsequent layers. The table in Figure 4a shows the color buffer, the normal buffer, the depth buffer and the edge maps of consecutive layers.

5.4. Layered Image Composition

For each layer the edge map and the color buffer are combined. Finally, the consecutive edge-enhanced layered images are blended into the frame buffer, which generates edge-enhanced order-independent transparent illustrations that provide a clear insight into the CSG model (Fig. 5a). Here, the edge intensities outline design and spatial assembly and the color shading provides additional spatial cues. To preserve a correct depth behavior with respect to the z-buffer, the z-values of a layer's depth buffer are considered for layered image composition.

6. Sketchy Illustrations

Sketchy illustrations of the design and spatial assembly can be implemented by adding uncertainty to each layers of the graphical decomposition. That is, the noise texture is applied to perturb the texture coordinates

of the edge maps and the color buffers, resp. shade maps before the final image composition. Combining the perturbation of the shade map (Fig. 4b) and the perturbation of the edge map (Fig. 4d) then results in a sketchy illustration for each layer. These sketchy illustrations are then blended in depth-sorted order into the frame buffer. Once again, the correct depth is ensured by also perturbing the depth buffer of each layer (Fig. 4c) before image composition.

5.2. Image Composition using Sketched Layers

As long as the same noise texture is used for all textures derived from all the layers, consecutive layers can still be blended into the frame buffer without disturbing artifacts, such as diverging edges or interpenetrating layers.

Consider one and the same texel location of the textures of consecutive layers. The corresponding texture value is accessed by specific texture coordinates (s,t) sampling that location in all textures regardless of whether the texture coordinates result from a previous perturbation or not. Figure 5a illustrates that the edge intensities of edge maps of consecutive layers match one another even though they have been perturbed. Consequently, the ordering of z-values specified by the ordering of the layers can be ensured for composing all sketchy illustrations.

Finally, the design decisions for composing CSG models can be communicated by illustrating their complex aggregation in a sketchy manner (Fig. 5b).

Conclusions

We have represented a novel interactive illustration technique that sketches the spatial assembly and the design of CSG models. It can be used a new tool for communicating and presenting CSG models in interactive environments.

We believe that the sketchy illustrations of the CSG models' layouts could encourage active participation, discussions, and reconsideration when designing these models. Though, a confirming precise user study is still required.

A major drawback of the presented technique is that visual cues such as dashed or dotted line styles for occluded edges can not be produced because the image-space approach does not provide an analytic representation for the edges so that individual line styles can not be supported.

References

[1] Diefenbach P. J. Pipeline Rendering: Interaction and Realism Through Hardware-based Multi-Pass Rendering. PhD Thesis. University of Pennsylvania, 1996.

[2] Everitt C. Interactive Order-Independent Transparency. Technical report. NVIDIA Corporation, 2001.

[3] Goldfeather J., Molnar S., Turk G., and Fuchs H. Near Realtime CSG Rendering Using Tree Normalization and Geometric Pruning, IEEE Computer Graphics and Applications, 9, 3, 20-28, 1989.

[4] Gooch, A., Gooch, B., Sousa, M. C. Illustrative Visualization. A. K. Peters. to appear.

[5] Gooch, B., Gooch, A. Non-Photorealistic Rendering. A. K. Peters. 2001.

[6] Isenberg, T.: Capturing the Essence of Shape of Polygonal Meshes, PhD Thesis, University of Magdeburg, 2003.

[7] Isenberg T., Freudenberg B., Halper N., Schlechtweg S., and Strotthotte T. A Developer's Guide to Silhouette Algorithms for Polygonal Models. IEEE Computer Graphics and Applications, 23, 4, 28-37, 2003.

[8] Kirsch F. Entwurf und Implementierung eines computergraphischen Systems zur Integration komplexer, echtzeitfähiger 3D-Renderingverfahren. PhD Thesis, University of Potsdam, 2005.

[9] Kirsch F. and Döllner J. Rendering Techniques for Hardware-Accelerated Image-Based CSG. Journal of WSCG'04, 221-228, 2004.

[10] Kilgard M. (Ed.). NVIDIA OpenGL Extension Specifications. NVIDIA Corporation, 2005.

[11] Koenderink, J. J.: What does the Occluding Contour tell us about Solid Shape?, Perception 13, pp.321-330, 1984.

[12] Nienhaus M. and Döllner J. Edge Enhancement – An Algorithm for Real-Time Non-Photorealistic Rendering. Journal of WSCG'03, 346-353, 2003.

[13] Nienhaus M. and Döllner J. Blueprints – Illustrating Architecture and Technical Parts using Hardware-Accelerated Non-Photorealistic Rendering. Proceedings of Graphics Interface 2004, 49-56, 2004.

[14] Nienhaus M. and Döllner J. Sketchy Drawings. Proceedings of ACM AfriGraph, pp. 73-81, 2004.

[15] Nienhaus M., Kirsch, F. and Döllner J. Illustrating Design and Spatial Assembly of Interactive CSG. Proceedings of ACM AfriGraph, pp.91-98, 2006.

[16] Perlin, K. An image synthesizer, Computer Graphics, 19(3), (Proceedings of ACM SIGGRAPH'85), pp.287-296, 1985.

[17] Rheingans P. and Ebert D. Volume Illustration: Non-Photorealistic Rendering of Volume Models. IEEE Transactions on Visualization and Computer Graphics 2001, 7, 3, 253-264, 2001.

[18] Rost R. T. OpenGL Shading Language. Addison-Wesley, 2004.

[19] Saito T. and Takahashi T. Comprehensible Rendering of 3-D Shapes. Computer Graphics (Proceedings of ACM SIGGRAPH'90), ACM, 197-206, 1990.

[20] Sousa M. S. and Ebert D. S. Computer-Generated Medical, Technical, and Scientific Illustration, ACM SIGGRAPH'05 Course, 2005.

[21] Schumann, J., Strothotte, T., Raab, A., and Laser, S. Assessing the Effect of Non-photorealistic Rendered Images in CAD. Proceedings of ACM SIGCHI 1996, 35-41, 1996.

[22] Strothotte, T., Masuch, M., and Isenberg, T. Visualizing Knowledge about Virtual Re-constructions of Ancient Architecture. Proceedings of CGI 1999, 36-43, 1990.

[23] Strothotte, T. and Schlechtweg, S. Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation, Morgan Kaufman, San Francisco, 2002.

[24] Wiegand T. F. Interactive Rendering of CSG Models. Computer Graphics Forum 1996, 15, 4, 249-261, 1996.